Multiple Clone Row DRAM: A Low Latency and Area Optimized DRAM

Jungwhan Choi* Wongyu Shin* Jaemin Jang* Jinwoong Suh*
Yongkee Kwon+ Youngsuk Moon+ Lee-Sup Kim*
*Korea Advanced Institute of Science and Technology +SK hynix
{cjwdream,wgshin,jmjang,jinwoongsuh}@mvlsi.kaist.ac.kr
{yongkee.kwon,youngsuk.moon}@sk.com leesup@kaist.ac.kr

Abstract

Several previous works have changed DRAM bank structure to reduce memory access latency and have shown performance improvement. However, changes in the area-optimized DRAM bank can incur large area-overhead. To solve this problem, we propose Multiple Clone Row DRAM (MCR-DRAM), which uses existing DRAM bank structure without any modification.

Our key idea is Multiple Clone Row (MCR), in which multiple rows are simultaneously turned on or off to consist of a logically single row. MCR provides two advantages which enable our low-latency mechanisms (Early-Access, Early-Precharge and Fast-Refresh). First, MCR increases the speed of the sensing process by increasing the number of sensed-cells. Thus, it enables a READ/WRITE command to an MCR to be issued earlier than possible for a normal row (Early-Access). Second, DRAM cells in an MCR exhibit more frequent refreshes without additional REFRESH commands, thereby reducing the amount of charge leakage during the refresh interval for the identical cell. The reduced amount of charge leakage enables a PRECHARGE command to be served before the activated-cells are fully restored (Early-Precharge) and a REFRESH operation to be completed before the refreshed-cells are fully restored (Fast-Refresh).

Even though MCR-DRAM sacrifices memory capacity for low-latency, it can be dynamically reconfigured from low-latency to full-capacity DRAM. MCR-DRAM improves both performance and energy efficiency for both single-core and multi-core systems.

1. Introduction

As the rate of improvement in processor now exceeds that of DRAM, DRAM performance is becoming a critical issue [31]. DRAM performance is mainly affected by two factors: bandwidth and latency. DRAM bandwidth has been increased fourfold by memory interface technology developments in the last 13 years. Meanwhile, DRAM latency, which is estimated by *tRC*, has been reduced by only about 11% [6, 28]. It is possible to reduce access latency by modifying the DRAM bank structure. However, DRAM vendors are reluctant to change the existing area-optimized DRAM bank. This is because the DRAM bank is the densest region in DRAM and has a very repetitive pattern, and so a small change of DRAM bank can incur significant area-overhead.

Recent DRAM architecture studies have concentrated on reducing DRAM latency and mitigating the cost increase [29, 13]. Previously, CHARM [29] partially reduces the number of rows per mat and adds inter-bank dataline wires to obtain low-latency, thereby incurring the area-overhead penalty. In another approach, TL-DRAM [13], isolation transistors were inserted between the far and near segment bitlines for low-latency. However, adding the transistors to sub-arrays increases area-overhead. CHARM and TL-DRAM are good proposals to enable low-latency DRAM and show high system performance improvements, but the modifications of the DRAM bank are still burdens to vendors because of the cost-sensitive DRAM market.

The main objective of our work is to make a low-latency DRAM that maintains the conventional DRAM bank structure for mass production. We briefly introduce our key observations and mechanisms (*Early-Access*, *Early-Precharge* and *Fast-Refresh*) to reduce latency. Then, we apply our key idea to solve the problems caused by the observations and enable low-latency mechanisms.

Key Observation 1. The sensing process of a cell with high capacitance (C_{cell}) is faster.

$$\Delta V = \frac{V_{DD}}{2}\frac{C_{cell}}{C_{cell}+C_{bit}} = \frac{V_{DD}/2}{1+C_{bit}/C_{cell}} \quad (1)$$

The above equation (1) shows a *charge sharing voltage* ΔV between a DRAM cell capacitor (C_{cell}) and bitline capacitor (C_{bit}) [19]. From the equation, the ΔV can be increased by high cell capacitance (C_{cell}). As explained in [27], increased ΔV can make the sensing process fast[1], *allowing column access (READ/WRITE) commands to be issued early (Early-Access)*. However, increasing the size of a DRAM cell to obtain high C_{cell} enlarges the area of DRAM cells, which is in conflict with the effort of DRAM vendors to reduce cost-per-bit.

[1] Increased ΔV can reduce time to reach certain voltage level.

ISCA '15, June 13 - 17, 2015, Portland, OR, USA

DOI: http://dx.doi.org/10.1145/2749469.2750402

Key Observation 2. *A short refresh interval for the identical cell reduces the amount of charge leakage of the cell.*

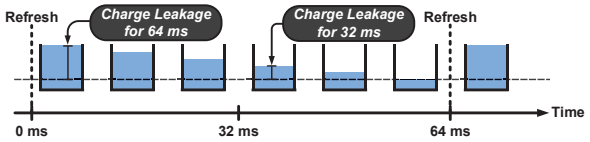

Figure 1: Charge leakage of a DRAM cell

Typically, a cell is refreshed every 64 ms to prevent loss of its data over time due to the charge leakage. Fig. 1, which uses an analogy to describe the charge (water) in a cell (water-tank), shows that the amount of charge leakage for 32 ms is less than that for 64 ms. Thus, the maximum amount of charge leakage of cells refreshed every 32 ms is smaller than that of cells refreshed every 64 ms.

Before data can be read, the data first must be loaded into sense-amplifiers (activation). During that process, the activated-cells initially lose and then gradually restore their data. To prevent data loss caused by the charge leakage until the next REFRESH/ACTIVATE command for the cells (maximum 64 ms), a PRECHARGE command should be served only after the activated-cells are fully restored. On the other hand, if the amount of charge leakage of the cells is reduced by a shorter cell refresh interval, *a PRECHARGE command can be served before the activated-cells are fully restored (Early-Precharge)*. In addition, *not fully refreshing the DRAM cells is allowed due to the reduction in the amount of charge leakage during the interval (Fast-Refresh)*. However, issuing additional REFRESH commands to reduce the amount of charge leakage may incur performance degradation and power loss.

As previously mentioned, achieving low-latency by using the observations can incur problems (larger DRAM cell size and additional REFRESH commands). We suggest our key idea to solve the problems and enable our low-latency mechanisms without DRAM bank modification.

Our key idea is to treat multiple rows as a logically single row, which enables the multiple rows to have the same data. We call a set of these multiple rows a Multiple Clone Row (MCR). The characteristic of the MCR is that every row in the MCR is simultaneously turned on or off. Due to this characteristic, high cell capacitance (C_{cell}) is achieved by multiple sensed-cells without new area-overhead and cells in MCRs have shorter cell refresh intervals without additional REFRESH commands. The detailed explanation is in Sec. 3.

MCR-DRAM has normal rows with existing-latency and MCRs with low-latency. As an MCR is a logical group of multiple rows, users can change the number of rows comprising an MCR (K*x* MCR), the ratio of MCR region to total row region (L*%reg*), and the Refresh-Skipping ratio (M/K*x*) (see Table 1 and Sec. 4.3 for the details).

Our work makes key contributions as follows:

- We introduce our observations that higher DRAM cell capacitance and the smaller amount of charge leakage during the same cell refresh interval enable DRAM latency to be reduced. To exploit our observations and solve the problems (larger DRAM cell size and additional REFRESH commands) caused by the observations, we suggest our key idea (MCR): a set of multiple rows considered as a logically single row. Our low-latency mechanisms (Early-Access, Early-Precharge and Fast-Refresh) are applied to MCRs. This enables MCRs to have the relaxed timing constraints (the reduction of *tRCD*, *tRAS* and *tRFC*).
- MCR-DRAM can be implemented by slightly modifying a peripheral region. Thus, MCR-DRAM does not incur additional area-overhead in a DRAM bank, which is the densest region.
- MCR-DRAM provides an MCR-mode, which is dynamically changed from low-latency mode to full-capacity mode. To enable MCR-mode to be dynamically reconfigurable, we exploit existing registers (MR) and commands (MRS). To the best of our knowledge, there has been no previous low-latency DRAM which is dynamically reconfigurable by users.
- We obtained the timing constraints for use with a variety of MCR-mode configurations by circuit level SPICE simulations. The SPICE simulations were performed based on 55 nm DRAM technology information [21]. Based on the timing constraints, we evaluated our MCR-DRAM using the system level simulation and showed that our proposal improved performance and energy efficiency for both single-core and multi-core simulations.

2. Background

2.1. DRAM Structure

Fig. 2 shows the conventional DRAM structure. As shown in the figure, a rank consists of multiple DRAM chips which work in union. A bank in a chip is composed of a lot of sub-arrays, each of which is made up of many mats. A mat is composed of an array of sense-amplifiers (row-buffer), wordline-drivers and a cell-array (512 cells by 512 cells). As the bit-per-area of a mat is one of key factors that decide cost-per-bit, mats are designed to be the densest region in DRAM. Furthermore, such dense mats are located in a bank repeatedly. Thus, a small modification of DRAM bank, especially of DRAM mat, for low-latency may incur large area-overhead compared to the change of the peripheral region. That is why the DRAM vendors are unwilling to modify the DRAM bank design.

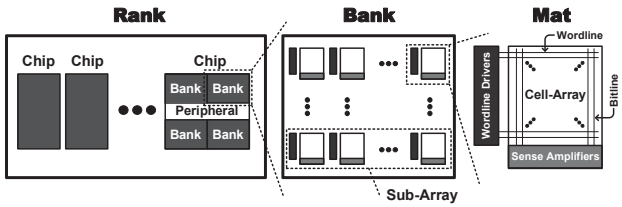

Figure 2: DRAM structure

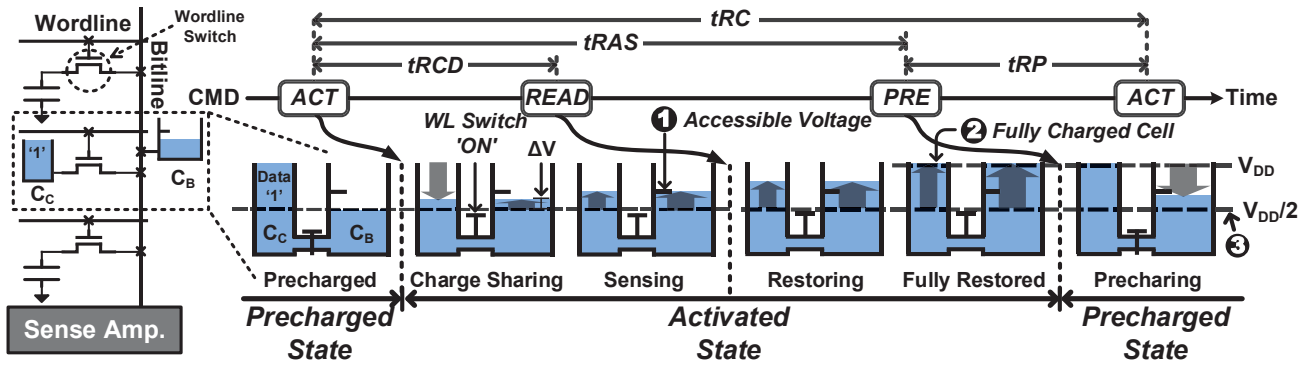

Figure 3: A CMD sequence and the corresponding voltage change of the cell and bitline

2.2. DRAM Operation and Timing Constraints

In Fig. 3, we explain DRAM internal operation and timing constraints in detail using a water model. The water model describes the charge-flow between the DRAM cell and the bitline capacitor. The voltage and capacitance are described as the level of water and the size of water-tank, respectively.

- **Precharged State.** A DRAM bank enters the precharged state by serving a PRECHARGE command. In this state, the row-buffer in the bank has no data. This state can be considered the 'ready' stage for the loading of the entire data of a row by an ACTIVATE command. For DRAM cells to be disconnected from the sense-amplifiers, all of the wordlines and bitlines in the bank are at 0V and $V_{DD}/2$, respectively.
- **Charge Sharing Stage of Activated State.** Upon receiving an ACTIVATE command and the target row address, the voltage of wordline corresponding to the row address is raised to V_{PP} (WL Switch 'ON') and the charge sharing stage starts as shown in Fig. 3. Depending on the data ('1' or '0') in the cell, the charge sharing voltage becomes ΔV for '1' or -ΔV for '0'. It is known that the magnitude of ΔV has an impact on the speed of the sensing process [27].
- **Sensing Stage of Activated State.** After the charge sharing stage, the row-buffer (sense-amplifiers) amplifies the small ΔV in the same direction. An access (READ/WRITE) command can be issued after the bitline voltage reaches the accessible voltage (❶), which is the minimum voltage to latch the accessed-data in the row-buffer. Accordingly, *tRCD*, the timing constraint to ensure the minimum time from an ACTIVATE to a READ/WRITE command, is decided by when the bitline voltage reaches ❶.
- **Restoring & Fully Restored Stage of Activated State.** As the data in the activated-cells are destroyed in the charge sharing stage, they should be recovered. This process is continued until the voltage in the activated-cell arrives at V_{DD} or 0V. A PRECHARGE command can be issued only *tRAS* after issuing an ACTIVATE command, which ensures the activated-cells are fully charged (❷). Therefore, the value of *tRAS* depends on when the cell-voltage reaches V_{DD} or 0V.
- **Precharing Stage of Precharged State.** By serving a PRECHARGE command, the raised-wordline voltage (V_{PP}) is lowered to 0V (WL Switch 'OFF') and the bitline voltage goes down (data '1') or up (data '0') to $V_{DD}/2$. *tRP* is the timing constraint for the minimum time from a PRECHARGE to an ACTIVATE command to guarantee the time for the bitline-voltage to reach $V_{DD}/2$ (❸).

2.3. DRAM Refresh

Because of the charge leakage characteristic, a DRAM cell should be refreshed every 64 ms (32 ms for high temperature) to maintain data integrity. This means all of the rows in a DRAM should be refreshed within 64 ms (or 32 ms).

Upon receiving a REFRESH command from the memory controller, the DRAM generates addresses of rows to be refreshed using internal refresh counter and refreshes all banks according to the generated row address. During the refresh process, the entire data from the target row are brought into the row-buffer and then written back to the original row, which is essentially identical to the *activation and precharge process* explained in Fig. 3. Thus, the time taken to refresh a row is *tRC* (*tRAS* + *tRP*).

According to JEDEC[2] standards [7], 8K REFRESH commands should be issued for 64 ms. As a DRAM bank has more than 8K rows in today's DRAM, multiple rows in each bank should be refreshed by a REFRESH command. This time taken to serve a REFRESH command is referred to as *tRFC*, the length of which is decided by the number of rows refreshed by the REFRESH command.

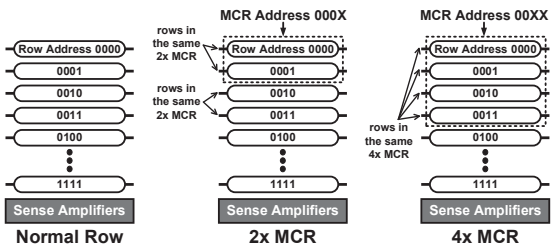

Figure 4: Concept of Multiple Clone Row

[2]Joint Electron Device Engineering Council

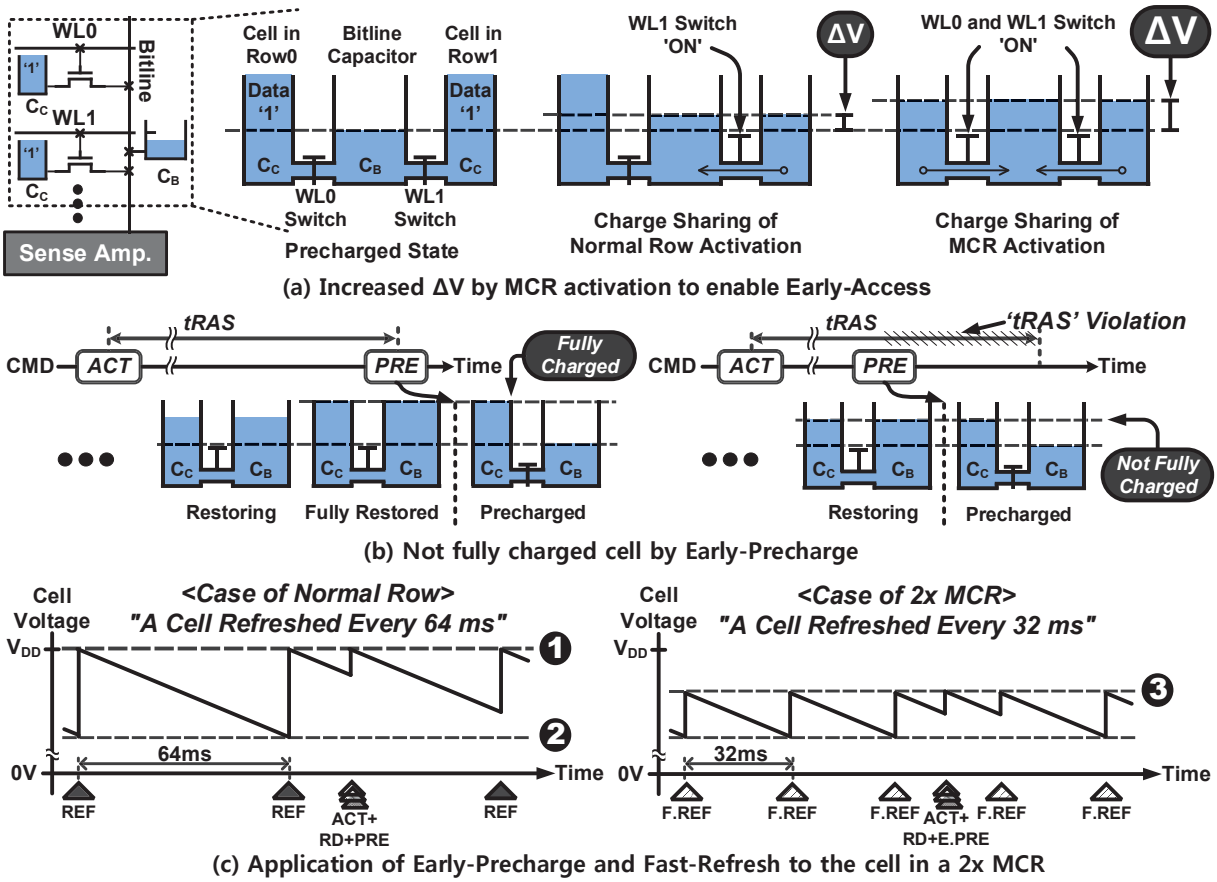

Figure 5: Three mechanisms to reduce access latency: Early-Access, Early-Precharge and Fast-Refresh

3. Key Concepts

3.1. Multiple Clone Row

K*x* MCR, MCR Address and Normal Row. An MCR is composed of multiple rows which are turned on or off simultaneously to be a logically single row. As shown in Fig. 4, an MCR consisting of K (2 or 4) rows is called K*x* MCR. Each K*x* MCR has its own address, which is made by ignoring $\log_2 K$-bit LSBs of the row addresses. For example, each 4*x* MCR is constructed by 4 consecutive rows in the same sub-array by ignoring 2-bit LSBs. When indicating an MCR address, we use 'X' to express ignored-bits of row addresses. Assuming 4-bit row address system, the MCR address 00XX means the MCR is composed of 4 consecutive rows (0000, 0001, 0010 and 0011). Unless otherwise specified, a row which is not a member of an MCR is called a normal row.

K Times Refreshed Cells in a K*x* MCR. As mentioned in the background section (Sec. 2.3), DRAM refreshes all of the rows for 64 ms. When a row in a K*x* MCR is refreshed, the other rows in the same K*x* MCR are concurrently refreshed. In this way, as a K*x* MCR has K rows, every row in the K*x* MCR is refreshed K times for 64 ms without issuing additional REFRESH commands, thereby reducing the refresh interval for the identical cell (or row). As a result, *the maximum amount of charge leakage*[3] of cells in the MCR is smaller than that of cells in normal row refreshed once for 64 ms.

3.2. Early-Access (Reduction of *tRCD*)

Fig. 5(a) shows the difference in charge sharing voltage between normal row activation and MCR activation. At the precharged state in Fig. 5(a), the bitline capacitor is precharged to have a voltage of $V_{DD}/2$, which is a ready state to obtain ΔV after the charge sharing stage when prompted by an ACTIVATE command. In the normal row activation, by turning on the wordline1 switch, the charge sharing process occurs between the cell in row1 and the bitline capacitor. Meanwhile, in the MCR activation, by turning on both the wordline0 and wordline1 switch, the magnitude of ΔV after the charge sharing stage is larger than that of normal row activation. As the increased ΔV speeds the sensing process [27], the activation of MCR causes the voltage of the bitlines to reach accessible voltage faster. Thus, MCR enables Early-Access (relaxed *tRCD*) by increasing the number of cells participating in a charge sharing process without additional area-overhead in banks.

[3]The exact amount of charge leakage cannot be anticipated because of PVT variations. However, assuming the worst case of DRAM cell, the maximum amount of charge leakage can be decided by the refresh interval for the identical cell.

3.3. Early-Precharge (Reduction of *tRAS*)

As shown at the left of Fig. 5(b), a PRECHARGE command should be served after the activated-cells are fully charged to ensure data integrity considering the amount of charge leakage for maximum 64 ms. On the other hand, at the right of Fig. 5(b), a PRECHARGE command is issued with a '*tRAS*' violation. In this situation, the corresponding wordline switch is turned off and the activated-cell is detached from the sense-amplifier, causing the voltage of the activated-cell to remain not fully charged.

Unlike Early-Precharges for normal rows, Early-Precharges for MCRs can make no data loss. For example, cells in normal rows are refreshed every 64 ms. Assuming that $0.2V_{DD}$ is the corresponding voltage drop caused by charge leakage for 64 ms, $0.8V_{DD}$ ($= 1.0V_{DD} - 0.2V_{DD}$) is logically allowable minimum cell voltage for correct data '1'. However, as cells in 2x MCRs can be refreshed every 32 ms due to more frequent refresh operations as explained in Sec. 3.1, $0.1V_{DD}$ is the corrpesonding voltage drop[4] for 32 ms. This means that even though cells in MCRs are charged up to $0.9V_{DD}$ ($= 0.8V_{DD} + 0.1V_{DD}$) by Early-Precharge, cells in MCRs maintain data integrity.

3.4. Fast-Refresh (Reduction of *tRFC*)

As the internal refresh operation for a row is the same as the *activation and precharge process*, the Early-Precharge mechanism can be applied to a refresh operation for the purpose of not fully refreshing the cells in MCRs. This MCR refresh operation which resembles *activation and Early-Precharge process* is called Fast-Refresh. At the left of Fig. 5(c), the DRAM cell is fully charged by a REFRESH command every 64 ms or ACTIVATE+(READ)+PRECHARGE commands (❶). The cell falls down to the data retention voltage (❷), which is logically considered as the minimum voltage-boundary of data '1' as explained in Sec. 3.3. For the case of the cell in a 2x MCR refreshed every 32 ms (right), even though the DRAM cell is not fully charged by Fast-Refresh every 32 ms or ACTIVATE+(READ)+Early-Precharge commands (❸), the voltage of the cell never falls below ❷.

4. Multiple Clone Row DRAM

4.1. MCR-Mode

MCR-Mode Configuration. In the above section, we showed that Early-Access, Early-Precharge, and Fast-Refresh for MCR is available. When MCR-mode is turned on, MCRs are allocated to the rows near the sense-amplifiers of each sub-array, as shown in Fig. 6. When full-capacity is necessary, MCR-DRAM can be used as a conventional DRAM by turning off the MCR-mode.

The MCR-mode provides a variety of modes. The detailed explanation is in Table 1, where 'M/K*x*' refers to Refresh-Skipping which will be explained in Sec. 4.3. Table 1 will also be used to explain our work.

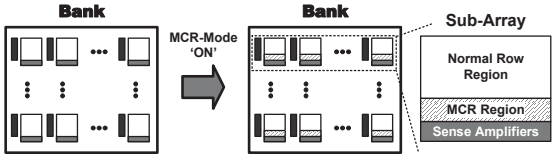

Figure 6: Location of MCRs

Table 1: MCR-mode configuration

MCR-Mode		*Description*
mode [M/Kx/L%reg]	*Kx*	'K' indicates the number of rows consisting of an MCR.
	M/Kx	'M' means the number of refresh operations in each Kx MCR for 64ms, (1 ≤ M ≤ K).
	L%reg	'L%' indicates the ratio of rows in all MCRs to all rows.

A Mode Register Set Command for MCR-Mode. In existing DRAMs, with programmed via Mode Register Set (MRS) commands, Mode Registers (MR) are used to define the specific mode of operation. An existing MRS command can be modified in order to enable an MCR-mode to be dynamically changed. By receiving an MRS command for MCR-mode from the memory controller, the corresponding MR[5] stores information in Table 1. Then, the MR transfers the information to our MCR generator, a role of which is to generate an MCR address from an original row address according to the MCR-mode. The detailed explanation is in Sec. 4.2.

4.2. Implementation of MCR

A Low-Cost Method to Generate an MCR Address. Fig. 7(a) shows a physical row decoding path and Fig. 7(b) is the corresponding simplified logical view. As shown in Fig. 7(b), an address received through the address pin is stored in the address buffer, where the internal address is generated according to the stored address. The internal address is composed of not only *N*-bit A (original address) bus also *N*-bit /A (inverted address). As shown in Fig. 7(b), each wordline is driven by N inputs, each *m*-th input of which is connected to either A_m or $/A_m$ ($0 \leq m \leq N-1$). From this fact, *Kx MCR can be implemented by assigning the logic 'high' to each of* $\log_2 K$*-bit LSBs of both A and* $/A$ ($A_{\log_2 K-1}...A_0 = 1...1$ and $/A_{\log_2 K-1}.../A_0 = 1...1$). For example, if the combination of internal addresses were $A_2A_1A_0 = 001$ and $/A_2/A_1/A_0 = 111$, both wordline0 and wordline1 would be driven by the logic 'high' inputs $/A_2/A_1/A_0$ and $/A_2/A_1A_0$, respectively. This enables two rows (address 000 and 001) to be members of the 2*x* MCR (MCR address 00X).

MCR Generator. As previously mentioned, to enable K*x* MCR, the $\log_2 K$-bit LSBs of the internal address (both A and /A) should be the logic 'high'. To make this possible, our MCR generator is added between the address buffer and the internal address line as indicated in Fig. 7(c). In Fig. 7(c), whenever the Mode Register for MCR-mode changes, the MCR generator receives new information regarding the

[4] We assume the voltage drop caused by charge leakage is proportional to the refresh interval for the identical cell.

[5] The bits reserved for future use such as A_{15}-A_3 of MR3 in DDR3 [7] can be used to enable dynamic MCR-mode change.

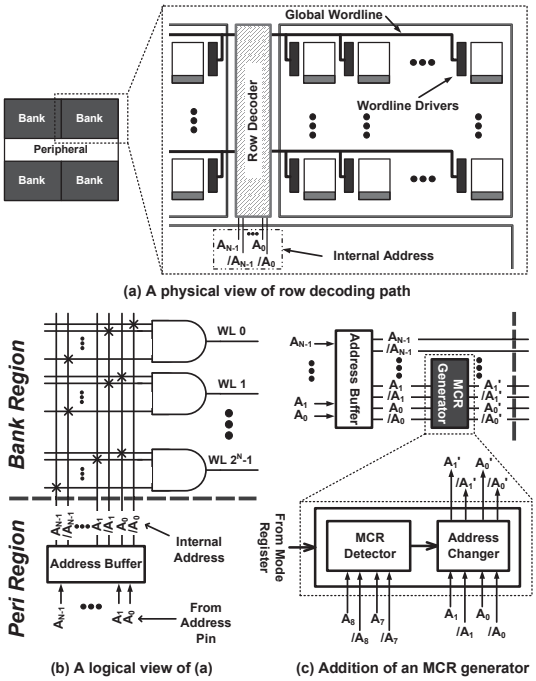

Figure 7: A row address path and an MCR generator

MCR-mode configuration in Table 1 from the Mode Register and then stores the information. When a row address comes in, the MCR detector first checks whether the row is in an MCR or not by using the information stored in the MCR detector. Assuming a DRAM with 512 rows in each sub-array, if the MCR region is assigned to 50% (mode [50%*reg*]) and 25% (mode [25%*reg*]) of all the rows, addresses of all rows in MCRs must contain $A_8 = 1$ and $A_8A_7 = 11$, respectively. In other words, whether the received row is in an MCR or not is decided by only 1 or 2 bits according to the MCR-mode configuration. After the MCR detection, if the row turns out to be a member of an MCR, the address changer changes the $\log_2 K$-bit LSBs of the row address to the logic 'high' to turn on every row in the same MCR. This MCR generator is also used for Fast-Refresh and Refresh-Skipping. Even though the additional circuit is required, its cost is negligible[6]. That is because the circuit can be implemented with only tens or hundreds of gates and is not located in a DRAM bank, but in a peripheral region.

Multiple Latency Memory Controller. To apply different timing constraints to MCRs, 2 bit comparator should be added in the memory controller to judge if an incoming request is MCR request or not. Furthermore, additional bits are necessary to check MCR requests in queues and to store values of MCR timing constraints.

4.3. Effective Use of Early-Precharge and Fast-Refresh

K to N-1-K Wiring: A Wiring Method to Maximize the Benefits from Early-Precharge and Fast-Refresh. As described at the left of Fig. 8(a), a DRAM chip generates refresh row addresses by using the internal refresh counter, the value of which is incremented whenever the DRAM chip receives a REFRESH command. After the counter reaches the maximum value, which is equivalent to the number of rows[7], the next value becomes zero. In Fig. 8(a), there are two kinds of wiring methods: K to K wiring (❶) and K to N-1-K wiring (❷). ❶ is a method that connects each K-th refresh counter bit (B_K) to the K-th row address bit (R_K). On the other hand, ❷ is used to connect each K-th refresh counter bit to the N-1-K-th row address bit, thereby causing the LSB of the row address to change last. Fig. 8(b) and (c) show the generated 3-bit refresh row addresses as wired by ❶ and ❷, respectively. Even though each refresh counter bit of (b) and (c) is incremented by '1' for each REFRESH command, the refresh row address of (c) is not equal to (b) because of the different wiring methods. We marked the same MCR address with black-colored background for each Kx MCR (000 for 1x, 00X for 2x and 0XX for 4x). In 1x MCR (normal row), both (b) and (c) have the same refresh interval for the identical normal row (64 ms). The refresh interval for the same MCR of both 2x and 4x MCR in (b) is not uniform, whereas those in (c) is uniform. The maximum amount of charge leakage of the cell is decided by the maximum refresh interval for the identical MCR (in (b), 56 ms for 2x and 40 ms for 4x; in (c), 32 ms for 2x and 16 ms for 4x). Accordingly, by applying ❷, the maximum amount of charge leakage can be minimized with no additional circuit.

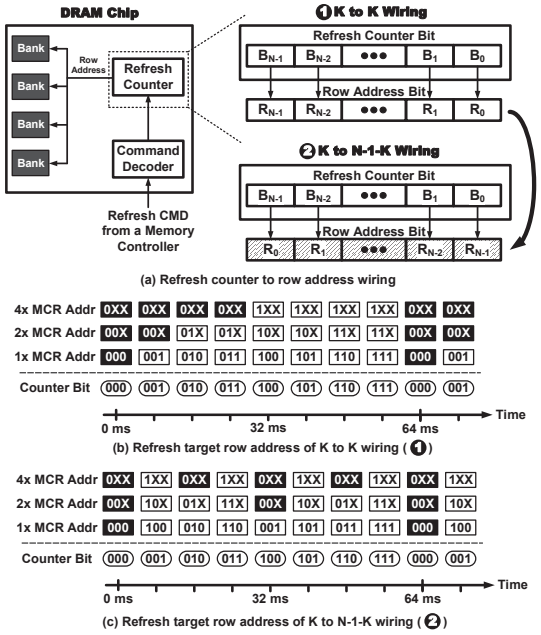

Figure 8: Two wiring methods and the corresponding refresh row addresses

Refresh-Skipping. Although the high refresh rate of high Kx MCR leads to low *tRAS* and *tRFC*, there is a possibility

[6]To lower circuit complexity, the MCR generator can be integrated with existing address buffer.

[7]Even though the refresh counter can be implemented with only $\log_2 8K = 3072$ bits, we explain the wiring methods assuming the counter bits equal to the row address bits for easy understanding.

to improve system performance and reduce power loss by not issuing a portion of all REFRESH commands. Fig. 9 shows the REFRESH commands for the same MCR. In Fig. 9, the 'M' of M/4x MCR means the number of refreshes in each 4x MCR for 64 ms. Furthermore, 'REF' and 'S' mean issuing and skipping the REFRESH command, respectively.

A 4/4x MCR (no Refresh-Skipping) is refreshed 4 times for 64 ms, each row in which is refreshed every 16 ms (64/4 ms). Meanwhile, as it has been refreshed 2 times for 64 ms, the row in 2/4x MCR has a higher refresh interval for the identical row (32 ms = 64/2 ms) compared to the 4/4x MCR. Although the 2/4x MCR has a higher *tRAS* than 4/4x MCR, additional commands can be issued during the Refresh-Skipping in mode [2/4x]. This occasionally improves performance and reduces power consumption in high capacity DRAM. To decide whether to skip the next REFRESH command or not, a refresh counter that counts up every REFRESH command should be added to the memory controller. Furthermore, to apply Fast-Refresh, that counter can be exploited to check if the next refresh row address is in an MCR.

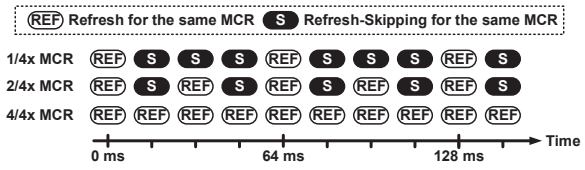

Figure 9: REFRESH commands for the same MCR according to the Refresh-Skipping ratio

4.4. Effective Use of MCR-DRAM

Prevention of Data Collision. Even though the MCR address is used to express the address of MCR in this work, computer systems use the original address. Thus, only one row in an MCR should be used because every row in the same MCR must have the same data. One of the methods to enable this is to allocate pages to only the first row in an MCR and make the other rows in the same MCR not page-allocated with the aid of the Operating System (OS). For example, assuming the mode [4/4x/100%reg], the OS can allocate pages to the rows with row address ...00 (first row in a 4x MCR). Then, page allocations to the other rows with row address ...01, ...10 and ...11 (the other rows in a 4x MCR) should be prohibited.

Profile-Based Page Allocation. To effectively use MCR-DRAM, the OS can allocate frequently-accessed pages to MCRs. In order to do the page allocation, the OS needs to obtain the information about frequently-accessed pages. The information is obtained with the aid of compiler-based or hardware-based profiling [29, 13]. As our work focuses on a new low-latency DRAM, a detailed page allocation mechanism is not provided in this work. Instead, for evaluations of our work, we use a pseudo profile-based page allocation method to reflect the allocation of frequently-accessed pages to MCRs. This method only allocates the frequently-accessed rows of each workload used in our single-core simulations to MCRs which are in the same bank (no change of channel, rank, bank and column address) in order not to exert an effect on bank-level parallelism and the row-buffer locality of workloads. In general, as the size of a page is smaller than that of a row, pseudo profile-based page allocation is a valid method[8].

Dynamic Change of MCR-Mode. If the capacity of the DRAM is sufficient, it is obvious that system performance is improved by low-latency of MCR-DRAM. However, if the capacity is deficient, the performance can be degraded by frequent page faults. As previously explained, the MCR-mode configuration can be dynamically changed using the MCR-mode MRS command. In a relaxed MCR-mode compared to the previous MCR-mode, it is possible to allocate pages to the rows in which page-allocation was previously forbidden. For example, when rows only with row addresses ...00 (first row in a 4x MCR) can be page-allocated in the previous mode [4/4x/100%reg], rows with row addresses ...00 or ...10 can be page-allocated in the new relaxed mode [2/2x/100%reg] (all rows when MCR-mode is turn off). In other words, the high Kx and/or L%reg MCR-mode can be dynamically changed to the low Kx and/or L%reg MCR-mode or turned off if performance degradation due to small capacity is predicted and the MCR-mode change makes no data collision.

An Effective Address Mapping to Prevent Data Collision and Enable Dynamic MCR-Mode Change. If only mode [100%reg] is used, data collision and dynamic MCR-mode change problems can be simply solved by locating R_0R_1 (R_k = k-th row address bit) to 2-bit MSBs of physical address as shown in Table 2. In the 4x MCR mode of Table 2, 2-bit MSBs of physical address cannot be used by making OS recognize only a quarter of the DRAM capacity. Then, the memory controller makes 2-bit MSBs of physical address '0' during address mapping, thereby enabling only first row to be accessed in a 4x MCR. When the mode is changed from 4x to 2x, the OS recognizes half of the original DRAM size and the memory controller makes 1-bit MSB of physical address '0', enabling only first row to be accessed in a 2x MCR.

Table 2: Physical address mapping to simply solve data collision and dynamic MCR-mode change problems

Allowed Mode Change	*MCR Mode*	*OS Recog.* Mem. Size	Physical Address Mapping	Accessible Row (... R_1R_0) ...11	...10	...01	...00
	4x MCR	N/4 GB	0 0 ●●●	X	X	X	O
	2x MCR	N/2 GB	0 R_1 ●●●	X	O	X	O
	Original	N GB	R_0 R_1 ●●●	O	O	O	O

Combination of 2x and 4x MCR. When the DRAM capacity is sufficient, to more effectively use the capacity, both 2x and 4x MCR can be used in an MCR-DRAM. In this mode, more/less frequently accessed pages are allocated to the 4x/2x MCRs, respectively.

4.5. Latency Analysis

MCR-DRAM has different timing constraints depending on the MCR-mode configuration. We conducted SPICE simula-

[8] A similar method is already applied in previous works [29, 13].

tions to obtain the timing constraints. For the SPICE simulations, the publicly available 55 nm DDR3 process technology [21] was used to obtain parameter information such as cell capacitance and bitline capacitance. Our simulation environment was scaled to meet timing constraints like *tRCD* and *tRAS*.

Fig. 10(a) shows the change of the bitline voltage after an ACTIVATE command. In the figure, as the number of rows in an MCR increases, the time taken to reach the accessible voltage level is reduced. As a result, the *tRCD* of 2*x* MCR and 4*x* MCR is reduced to 9.94 ns and 6.90 ns, respectively.

Fig. 10(b) indicates the voltage change of a cell after an ACTIVATE command. The *tRAS* of 1*x* MCR (normal row) is 35 ns. The restored voltage of the high K*x* MCR is initially high compared to that of the low K*x* MCR. However, as time goes on, the restoring speed of the high K*x* MCR is gradually slower than that of the low K*x* MCR. That is because the number of cells to be restored by a sense-amplifier is increased in the high K*x* MCR. In spite of the phenomenon, by applying Early-Precharge, the *tRAS* of 2*x* MCR and 4*x* MCR is reduced to 21.46 ns and 20.00 ns, respectively. The PRECHARGE voltage of 2*x* and 4*x* MCR is obtained by conservatively considering the advantage from the amount of charge leakage of a cell for 32 ms and 16 ms, respectively. Despite the conservative consideration, each *tRAS* of 2*x* and 4*x* MCRs is significantly reduced. This is because it takes such a long time to restore the last small amount of the cell voltage. Even though Fig. 10 shows data '1' case, it is known that DRAMs are designed to have almost the same timing constraints irrelevant to data values ('1' and '0').

The timing constraints of a variety of MCR-modes which are based on the SPICE simulation are summarized in Table 3.

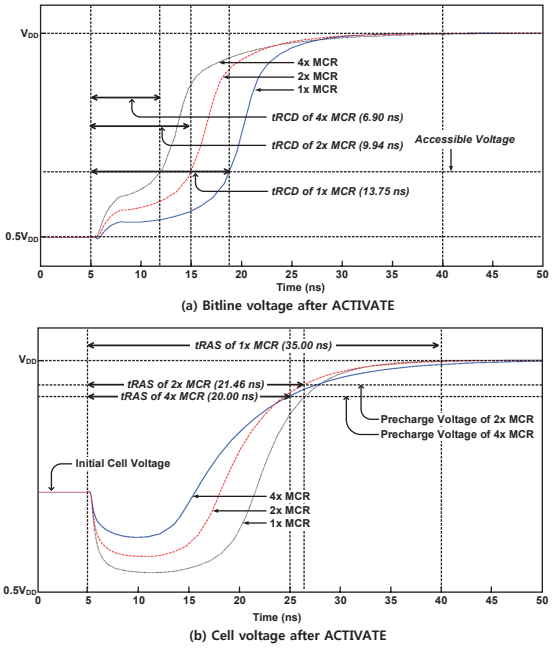

Figure 10: SPICE simulations of a variety of MCRs

Table 3: Timing constraints of a variety of MCRs

Timing Constraint	Device	1x MCR	2x MCR		4x MCR		
		1/1x	1/2x	2/2x	1/4x	2/4x	4/4x
tRCD (ns)	1 / 4 Gb	13.75	9.94	9.94	6.90	6.90	6.90
tRAS (ns)	1 / 4 Gb	35	37.52	21.46	46.51	22.78	20.00
tRFC (ns)	1 Gb	110	118.46	81.79	138.21	84.62	76.15
	4 Gb	260	280	193.33	326.67	200	180

5. Evaluation Methodology

5.1. Simulation Setting

We used USIMM [18, 30] from MSC (Memory Scheduling Championship) [1] as our system simulator. Based on the timing constraints in Table 3, the technical note from Micron [14] and Rambus power model [21], we modified USIMM to verify the improvement of performance and energy efficiency of the MCR-DRAM with the assumption that the DRAM capacity is sufficient.

Our baseline system configuration comprises the default USIMM single/quad core processor for single/multi core simulation, respectively, with the DRAM system which has 1 channel, 2 rank/channel, 8 banks/rank, and 32768 rows/bank (4 GB DRAM for single-core) or 131072 rows/bank (16 GB DRAM for multi-core) as shown in Table 4. In the memory controller, we modeled each read and write queue and used the address mapping policy [33, 26] as indicated in Table 4. The well-known FR-FCFS [23], which is appropriate for maximizing throughput, is used as our scheduling policy. The execution time/read latency improvement and energy-delay product (EDP) reduction are used for performance and energy efficiency evaluation, respectively.

Table 4: Baseline system configuration

Processor	Number of cores: 1 / 4, Clock speed: 3.2GHz, ROB size: 128, Retire width: 2, Fetch width: 4, Pipeline depth: 10
Memory Controler	Read queue capacity: 32, Write queue capacity: 32, Write queue high watermark: 24, Write queue low watermark: 8, Address mapping policy: page interleaving, Scheduling policy: FR-FCFS
DRAM	Memory bus speed: 800MHz, DDR3 memory channels: 1, Ranks per channels: 2, Banks per rank: 8, Rows per bank: 32768 / 131072, Columns (cache lines) per row: 128, DRAM size: 4 GB / 16 GB

5.2. Workloads

For our simulations, we employed the workloads used in MSC (Memory Scheduling Championship) [1]. As shown in Table 5, there are various workloads from 4 suites (COMMERCIAL, SPEC, PARSEC and BIOBENCH). In single-core simulations, we exploited each of the 16 workloads in Table 5 except for multi-threaded workloads ('MT-fluid' and 'MT-canneal'). To perform multi-core simulations, we used total 16 workloads, the 14 workloads among which were multi-programmed workloads made by randomly selecting single workloads from each of the 4 suites. The remaining 2 workloads are multi-threaded workloads.

To show the effect of the profile-based page allocation, the pseudo profile-based page allocation method was applied as described in Sec. 4.4. To make multi-programmed workloads for simulations on profile-based page allocation, we used the

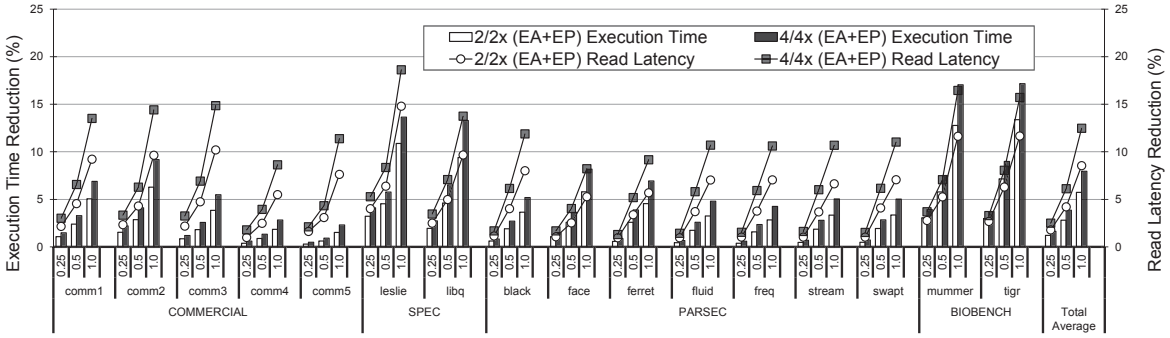

Figure 11: Single-core simulation according to the MCR ratio

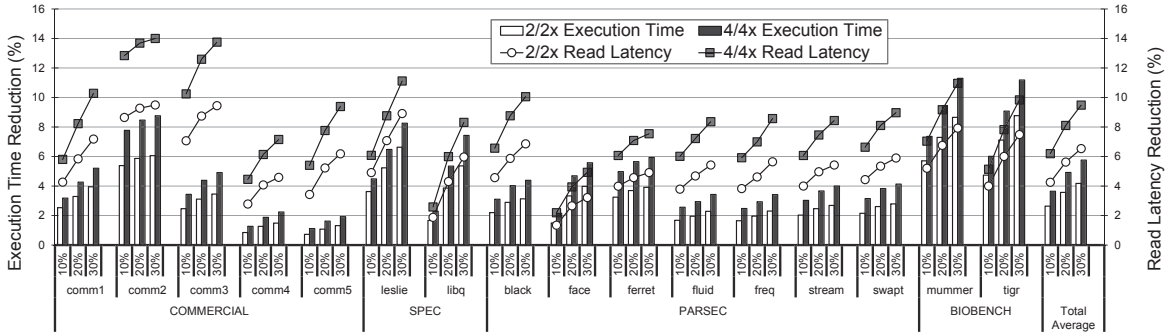

Figure 12: Effect of profile-based page allocation on single-core simulation

same way used to make the multi-programmed workloads (not for profile-based page allocation) except that we selected pseudo profile-based page allocated single workloads instead of original single workloads.

Table 5: Workloads

Suites	Workloads
COMMERCIAL	comm1, comm2, comm3, comm4, comm5
SPEC	leslie, libq
PARSEC	black, face, ferret, fluid, freq, stream, swapt, MT-canneal, MT-fluid
BIOBENCH	mummer, tigr

6. Results

6.1. Single-Core Simulation

Sensitivity to the MCR to Total Row Ratio. Unlike conventional DRAM, MCR-DRAM has asymmetric latency depending on whether the accessed-row is in an MCR or not, which is caused by Early-Access and Early-Precharge. Thus, to analyze the effect of the MCR ratio on performance, only Early-Access and Early-Precharge are applied (no Fast-Refresh and Refresh-Skipping). In addition, *tRCD* and *tRAS* of normal rows in each sub-array are changed to that of Kx MCRs depending on the MCR to total row ratio. In other words, the ratio of rows having *tRCD* and *tRAS* of Kx MCR to total rows in each sub-array is equal to the MCR to total row ratio. 0.25, 0.5 and 1.0 on the x-axis indicate the MCR to total row ratio.

Fig. 11 shows the performance improvement of the MCR-DRAM over the baseline. For all of the workloads, MCR-DRAM reduces execution time and read latency compared to the baseline. Because of the relaxed timing constraints of 4x MCR, mode [4/4x] shows better performance than mode [2/2x] for each of the MCR ratio. For both mode [2/2x] and mode [4/4x], performance is consistently improved with increasing MCR ratio. Mode [4/4x] reduces the execution time and read latency by up to 17.2% for 'tigr' and 18.6% for 'leslie', respectively. For mode [4/4x] with the 1.0 MCR ratio, on average, 7.9% reduction in execution time and 12.5% reduction in read latency are obtained.

However, although the used capacity of mode [4/4x] with 0.5 is higher that of mode [2/2x] with 1.0, mode [2/2x] with 1.0 shows better performance. While, on average, mode [2/2x] with 1.0 improves execution time by 5.7% and read latency by 8.5% over the baseline, mode [4/4x] with 0.5 only reduces execution time by 3.9% and read latency by 6.1%. This implies that, when the capacity is not sufficient, mode [2/2x] can perform better than the mode [4/4x].

Effect of Profile-Based Page Allocation. By applying profile-based page allocation, MCR-DRAM can achieve a high performance improvement with relatively small capacity loss. Fig. 12 shows the effect of profile-based page allocation on MCR-DRAM with mode [50%*reg*] (not the MCR to total row ratio). 10%, 20% and 30% on the x-axis indicate the pseudo profile-based page allocation ratio. Based on the allocation ratio, the frequently accessed rows are allocated to MCRs and the remainders are allocated to normal rows. Thus, regardless of the mode [50%*reg*], the number of requests for MCR is only affected by the pseudo profile-based page allocation ratio.

In Fig. 12, as the allocation ratio increases, execution time

and read latency are consistently reduced for all workloads. The execution time and read latency are improved by up to 11.3% for 'mummer' and 14.0% for 'comm2', respectively. However, the rate of performance improvement diminishes in general. This is because, in many workloads, addresses of memory requests are non-uniformly distributed[9].

MCR-Mode Analysis. Fig. 13 shows the impact of a variety of MCR-mode on performance. We use 10% pseudo page allocation, which enables the request ratio to MCR to be the same regardless of mode [L%*reg*]. In other words, performance improvement due to Early-Access and Early-Precharge is only affected by mode [M/K*x*]. Furthermore, mode [L%*reg*] only has an impact on Fast-Refresh and Refresh-Skipping.

With an increasing number of Refresh-Skippings for the same K*x* MCR, execution time improvements are consistently reduced in Fig. 13. This is because, even though more commands can be issued during Refresh-Skipping, the tight timing constraints of the MCR offsets the benefits from Refresh-Skipping. However, if DRAM system has high refresh power consumption, Refresh-Skipping can be an appropriate selection such as with mode [2/4*x*/75%*reg*] shown in Fig. 13. Compared to mode [4/4*x*/75%*reg*], mode [2/4*x*/75%*reg*] shows almost the same performance along with low refresh power consumption[10].

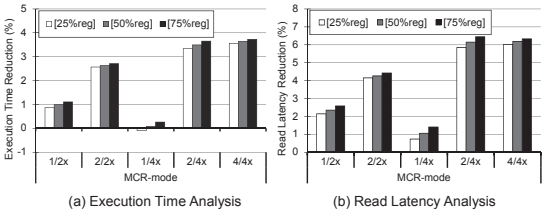

Figure 13: Effect of MCR-mode on single-core simulations

6.2. Multi-Core Simulation

To verify MCR-DRAM also improves performance in multi-core systems, we conduct the same analysis as in the single-core simulations.

Sensitivity to the MCR to Total Row Ratio. Fig. 14 presents multi-core simulation results according to the MCR to total row ratio. In the simulations, only Early-Access and Early-Precharge is used.

Mode [4/4*x*] performs better than mode [2/2*x*] for each of the MCR ratios because of the relaxed timing constraints of 4*x* MCR. On average, mode [4/4*x*] with 1.0 provides a 10.3% improvement in execution time and a 10.2% improvement in read latency. Also, although used capacity of mode [4/4*x*] with 0.5 is higher than mode [2/2*x*] with 1.0, mode [2/2*x*] with 1.0 performs better more than mode [4/4*x*] with 0.5. These trends are similar to the results of single-core simulations.

[9]In our extra analysis for 'comm2', there are 88.34% requests to MCRs and the remainders to normal rows for 10% pseudo profile-based page allocation.

[10]Our additional analysis indicates that the refresh power consumption of mode [2/4*x*/75%*reg*] is about 66.3% of that of mode [4/4*x*/75%*reg*].

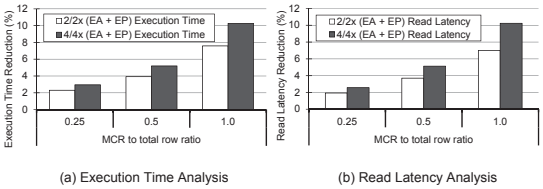

Figure 14: Multi-core simulations according to the MCR ratio

Effect of Profile-Based Page Allocation. In Fig. 15, as the page allocation ratio increases, execution time and read latency are consistently improved. On average, mode [4/4*x*/50%*reg*] with the 30% allocation ratio reduces execution time and read latency by 7.8% and 7.5%, respectively. As expected, with increasing page allocation ratio, the rate of performance improvement gradually diminishes.

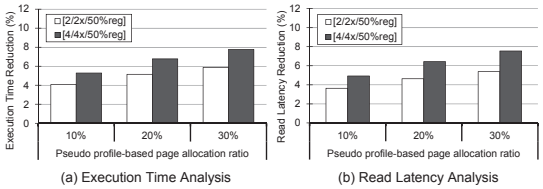

Figure 15: Effect of profile-based page allocation on multi-core simulations

MCR-Mode Analysis. In Fig. 16, 10% pseudo page allocation was applied. The difference of performance improvements between different mode [L%*reg*] (25%/50%/75%) is larger than single-core simulation results (Fig. 13). This is mainly because the effects of Fast-Refresh and Refresh-Skipping are increased due to high capacity DRAM (16 GB). Unlike the results in Fig. 13, mode [2/4*x*/75%*reg*] performs better than mode [4/4*x*/75%*reg*], which shows that Refresh-Skipping occasionally improves performance.

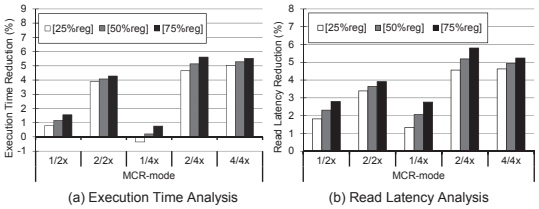

Figure 16: Effect of MCR-mode on multi-core simulations

6.3. Mechanism Analysis

Fig. 17 indicates the effect of mechanisms on both single-core and multi-core simulations. The baseline is existing DRAM and mode [100%*reg*] is applied to MCR-DRAM. The value in brackets indicates execution time reduction which is normalized to case 3. From Fig. 17, we draw a conclusion that Early-Access and Early-Precharge are the main reasons of performance improvement of MCR-DRAM. Furthermore, unlike cases 3 and 4, case 2 has the advantage in that an additional refresh counter is not required in the memory controller. In addition, in Fig. 17(a), case 4 shows a lower execution time improvement compared to case 2. This is because the negative effects of increased *tRAS* due to Refresh-Skipping

offset the performance benefit due to additional commands during Refresh-Skipping.

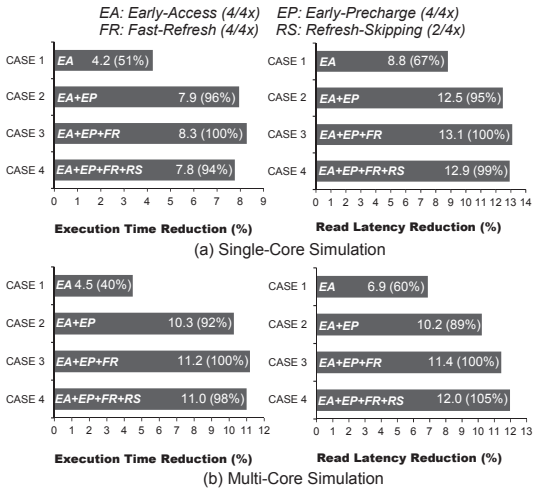

Figure 17: Effect of mechanisms

6.4. Energy Efficiency Analysis

When an MCR is activated, additional energy, which is relatively small compared to that of sense-amplifiers [21], is necessary to activate multiple wordlines. However, only a set of sense-amplifiers are used, just as in the normal row activation. Furthermore, the increase of activation and refresh energy consumption can be canceled out by not fully activating the cells, bitlines and sense-amplifiers (Early-Precharge and Fast-Refresh) and by Refresh-Skipping. Additionally, increased idle time by Early-Precharge and/or Refresh-Skipping can enable MCR-DRAM to operate in low-power mode for long time.

Fig. 18 shows the reduction of EDP of the MCR-DRAM over the baseline. The mode [4/4x/100%reg] shows the best EDP improvements for both single-core (14.1%) and multi-core (23.2%) simulations. In spite of Refresh-Skipping, mode [2/4x] improves EDP less than mode [4/4x]. This is because, in the simulations, the ratio of refresh energy consumption is not high enough for the EDP of mode [2/4x] to surpass that of mode [4/4x].

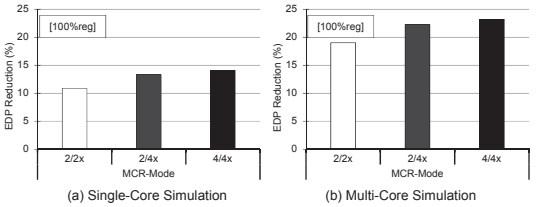

Figure 18: EDP improvements

7. Related Work

Low Latency DRAM. There are specialized DRAMs which reduce row-per-mat to obtain low-latency such as Fast Cycle DRAM (FCRAM) [24, 4] and Reduced Latency DRAM (RLDRAM) [15, 4]. However, these DRAMs induce large area-overhead of up to 80% [3, 29]. Row-buffer decoupling [20] adds isolation transistors in banks, which enables sense-amplifiers to keep the data even when the bitlines are precharged early. Adaptive-latency DRAM [12] exploits the extra amount of cell charge for the common case. As our work does not consider some effects from the common case, adaptive-latency DRAM can be exploited in our work.

Multiple Row Activation in a Bank. Kim et al. [9] suggested a method to exploit subarray-level parallelism. This enables multiple rows in different sub-array within a bank to remain to be activated for the purpose of overlapping of latency. Other works [5, 32] also enable multiple rows to be activated in a bank by activating only a portion of entire row. A patent [11] of IBM suggested a latched row decoder which activates multiple wordlines in order to reduce test time for defective-cell detection. However, in MCR-DRAM, even though multiple-rows are activated, only one set of sense-amplifiers is activated in a bank for the purpose of low-latency. RowClone [25] exploits back-to-back activation to copy data from one row to another row. Unlike it, our work simultaneously turns on or off multiple wordlines to make an MCR.

Exploiting of Charge Variation in DRAM Cells. Shin et al. [27] proposed a memory controller which improves performance by using the fact that the different amount of charge in cells leads to different access latency. In their work, the charge variation is anticipated by the amount of time elapsed after refresh. However, it is difficult to predict the amount of charge in cells because of PVT variations. Unlike it, in our MCR-DRAM, the timing constraints of MCR can be decided assuming the worst case of cells.

Low Latency Rows Used as Caches. Mechanisms to use the near-segment (low-latency row) from sense-amplifiers as caches to the far-segment in the same sub-array was introduced in TL-DRAM [13]. In our work, MCRs also can be used as caches to the normal rows in the same sub-array. Furthermore, unlike TL-DRAM, MCR-DRAM can provide a variety of cache sizes and latencies by MCR-mode change.

Memory Scheduling. Previous works [8, 10, 2, 16, 17] suggested memory scheduling methods to increase fairness and/or throughput. MCR-DRAM can achieve more system performance improvement in conjunction with those works because our work does not require a specific memory scheduling method.

Partial Access Mode. In Partial Access Mode (PAM) [22], one cell/bit is converted to multiple cells/bit to reduce the retention time variation among cells. As the refresh operation frequency is determined by a few weaker cells, the cell retention time which is extended by the variation reduction enables low refresh power. Even though our work stores the same data into multiple cells like PAM, MCR-DRAM focuses on reducing memory latency by exploiting low timing constraints of MCR. The timing constraints are reduced by Early-Access exploiting high charge sharing voltage of MCR and by Early-Precharge and Fast-Refresh exploiting the fact

that cells in MCRs need not be fully charged due to more frequent refreshes of MCR. Furthermore, as PAM needs copy operation, a circuit to enable not only simultaneous multiple wordline selection but also delayed wordline selection is added to the row decoder. However, our work does not require the delayed wordline selection, reducing circuit complexity of MCR-DRAM compared to PAM. In addition, we deal with a variety of architectural and/or high level issues and show some methods to effectively use MCR-DRAM.

8. Conclusion

Low access latency is important to improve computer system performance. However, modifications of bank structure are almost forbidden because of the cost-sensitive DRAM market. To our knowledge, the present study is the first work that enables low-latency DRAM without any bank modification.

We focused on two key observations: 1) the sensing process of DRAM cells with higher capacitance is faster, 2) cells with shorter refresh interval have the smaller amount of charge leakage during the interval. From these observations, we proposed Multiple Clone Row (MCR) to enable three mechanisms: Early-Access (Low *tRCD*), Early-Precharge (Low *tRAS*) and Fast-Refresh (Low *tRFC*).

To implement the MCR, we proposed an MCR generator which converts existing row addresses to an MCR address, just by modifying peripheral circuits. We also showed a wiring method to maximize the benefits from Early-Precharge and Fast-Refresh. Furthermore, Refresh-Skipping was suggested to reduce refresh energy consumption. MCR-DRAM provides an MCR-mode which can dynamically change latency, capacity and the Refresh-Skipping ratio. Thus, the MCR-DRAM also can be used like an existing full-capacity DRAM.

MCR-DRAM with mode [4/4*x*/100%*reg*] improves execution time/read latency/EDP by 8.3%/13.1%/14.1% in single-core simulations and by 11.2%/11.4%/23.2% in multi-core simulations on average.

MCR-DRAM is expected to be a solution to the conflict between latency and cost because it avoids the obstacle to area-optimization.

Acknowledgments

We would like to thank the reviewers for valuable feedback. This research was supported by SK hynix and by Basic Science Research Program through the NRF of Korea funded by Ministry of Science, ICT & Future Planning (NRF-2014R1A2A1A05004316).

References

[1] (2012) MSC:Memory Scheduling Championship. [Online]. Available: http://www.cs.utah.edu/~rajeev/jwac12/

[2] R. Ausavarungnirun, K. K.-W. Chang, L. Subramanian, G. H. Loh, and O. Mutlu, "Staged Memory Scheduling: Achieving High Performance and Scalability in Heterogeneous Systems," in *ISCA*, 2012.

[3] Brent Keeth, R. Jacob Baker, Brian Johnson, Feng Lin, *DRAM Circuit Design: Fundamental and High-Speed Topics*. John Wiley and Sons Inc., 2008.

[4] Bruce Jacob, Spencer W. Ng, David T. Wang, *Memory Systems(Cache, DRAM, Disk)*, 1st ed. Morgan Kaufmann, 2008.

[5] N. D. Gulur, R. Manikantan, M. Mehendale, and R. Govindarajan, "Multiple Sub-row Buffers in DRAM: Unlocking Performance and Energy Improvement Opportunities," in *ICS*, 2012.

[6] hynix. (2006) 256Mb Synchronous DRAM based on 4M x 4Bank x16 I/O. [Online]. Available: http://www.hynix.com/datasheet/pdf/dram/HY57V561620F(L)T(P)-xI(Rev0.2).pdf

[7] JESD79-3F, *"DDR3 SDRAM STANDARD"*. JEDEC, 2012.

[8] Y. Kim, M. Papamichael, O. Mutlu, and M. Harchol-Balter, "ATLAS: A scalable and high-performance scheduling algorithm for multiple memory controllers," in *HPCA*, 2010.

[9] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu, "A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM," in *ISCA*, 2012.

[10] Kim, Yoongu and Papamichael, Michael and Mutlu, Onur and Harchol-Balter, Mor, "Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior," in *MICRO*, 2010.

[11] T. Kirihata and H. Wong, "Latched row decoder for a random access memory," Patent, 1997.

[12] D. Lee, Y. Kim, G. Pekhimenko, S. Khan, V. Seshadri, K. Chang, and O. Mutlu, "Adaptive-latency DRAM: Optimizing DRAM timing for the common-case," in *HPCA*, 2015.

[13] D. Lee, Y. Kim, V. Seshadri, J. Liu, L. Subramanian, and O. Mutlu, "Tiered-latency DRAM: A low latency and low cost DRAM architecture," in *HPCA*, 2013.

[14] Micron. (2007) Calculating Memory System Power For DDR3. [Online]. Available: http://www.micron.com/-/media/documents/products/technical%20note/dram/tn41_01ddr3_power.pdf

[15] Micron Technology Inc., *RLDRAM3 Datasheet*, 2011.

[16] O. Mutlu and T. Moscibroda, "Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors," in *MICRO*, 2007.

[17] Mutlu, Onur and Moscibroda, Thomas, "Parallelism-Aware Batch Scheduling: Enhancing Both Performance and Fairness of Shared DRAM Systems," in *ISCA*, 2008.

[18] N. Chatterjee, R. Balasubramonian, M. Shevgoor, S. Pugsley, A. Udipi, A. Shafiee, K. Sudan, M. Awasthi and Z. Chishti, *"USIMM: the Utah Simulated Memory Module"*. University of Utah, Technical Report, 2012.

[19] D. H. Neil H.E. Weste, *CMOS VLSI Design. A Circuit and Systems Perspective* , 3rd ed. Addison-Wesley, 2005.

[20] S. O, Y. H. Son, N. S. Kim, and J. H. Ahn, "Row-buffer decoupling: A case for low-latency dram microarchitecture," in *ISCA*, 2014.

[21] Rambus. (2010) DRAM Power Model. [Online]. Available: http://www.rambus.com/energy

[22] Y. Riho and K. Nakazato, "Partial Access Mode: New Method for Reducing Power Consumption of Dynamic Random Access Memory," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, July 2014.

[23] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens, "Memory Access Scheduling," in *ISCA*, 2000.

[24] Y. Sato, T. Suzuki, T. Aikawa, S. Fujioka, W. Fujieda, H. Kobayashi, H. Ikeda, T. Nagasawa, A. Funyu, Y. Fuji, K. Kawasaki, M. Yamazaki, and M. Taguchi, "Fast Cycle RAM (FCRAM); A 20-ns Random Row Access, Pipe-Lined Operating DRAM," in *VLSI*, 1998.

[25] V. Seshadri, Y. Kim, C. Fallin, D. Lee, R. Ausavarungnirun, G. Pekhimenko, Y. Luo, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry, "RowClone: Fast and Energy-efficient in-DRAM Bulk Data Copy and Initialization," in *MICRO*, 2013.

[26] J. Shao and B. T. Davis, "The Bit-reversal SDRAM Address Mapping," in *SCOPES*, 2005.

[27] W. Shin, J. Yang, J. Choi, and L.-S. Kim, "NUAT: A non-uniform access time memory controller," in *HPCA*, 2014.

[28] SK hynix, *H5TQ2G63FFR*, 2013.

[29] Y. H. Son, S. O, Y. Ro, J. W. Lee, and J. H. Ahn, "Reducing Memory Access Latency with Asymmetric DRAM Bank Organizations," in *ISCA*, 2013.

[30] UTAH ARCH. (2012) USIMM: the Utah Simulated Memory Module. [Online]. Available: http://utaharch.blogspot.kr/2012/02/usimm.html

[31] W. A. Wulf and S. A. McKee, "Hitting the Memory Wall: Implications of the Obvious," *SIGARCH Comput. Archit. News*, March 1995.

[32] T. Zhang, K. Chen, C. Xu, G. Sun, T. Wang, and Y. Xie, "Half-DRAM: A high-bandwidth and low-power DRAM architecture from the rethinking of fine-grained activation," in *ISCA*, 2014.

[33] Z. Zhang, Z. Zhu, and X. Zhang, "A Permutation-based Page Interleaving Scheme to Reduce Row-buffer Conflicts and Exploit Data Locality," in *MICRO*, 2000.